The AIIDE-17 Workshop on
Intelligent Narrative Technologies
WS-17-20

Dynamic and Accelerated Partial Order Planning for Interactive Narratives

Xun Zhang,[1] Bhuvana C. Inampudi,[1] Norman I. Badler,[2] Mubbasir Kapadia[1]
[1]Computer Science Department, Rutgers University New Brunswick
[2]Computer and Information Science Department, University of Pennsylvania

Abstract

This paper explores new narrative generation paradigms for open world problems. We propose a speed-up variant of partial planner–accelerated partial order planner, that can automatically generate narratives for large plan spaces. To incorporate real-time free-form user interaction, a dynamic partial planning technique has been introduced to self-repair the narratives. We also propose a scalable and robust framework to craft open world narratives with minimal effort. Our approach enables content creators to craft complex open world narratives without explicitly authoring user interaction arcs. We tested our framework by developing multiple narratives with free-form interactions. Those narratives were used to test the robustness of the proposed planners.

Introduction

Partial order planners (POPs) are being used to generate complex narratives with a lot of user interaction. As the planning time increases with the number of actions, using POPs for open world narratives is not time efficient. Furthermore, if those worlds are to accommodate free-form user interaction, the plan should consider all the possible conflicts to the narrative. This, in turn, increases the plan space and effort required from content editors to craft a complete narrative.

The existing techniques either support narratives with little user interaction, fixed, or less branching narratives with a lot of user interactions. These approaches are not suitable for open world problems as the action space will be very enormous and including all user interactions will drastically increase the space complexity.

In this paper, we propose an approach that generates and repairs open world narratives in real time. This approach accommodates free-form user interactions in a generated narrative with minimum replanning. This approach can be used to recreate any historical event (for example, President William McKinley's assassination) and analyze how changes in various interactions would have affected that event or to generate interactive movies.

For such a system the planner should be able to generate plans in real time by using characters and actions that are relevant to that scene. The system should replan for any changes in the environment in real time, thereby accommodating free-form user interactions. To validate or use such a planner, the system architecture should be modular, robust and scalable to generate extensive narratives.

Though there hasn't been a single approach which tackles all the challenges, there has been attempts to address individual problems. Shoulson et al. (2011) introduced parameterized behavior trees to improve the modularity and reusability of behavior trees. Kapadia et al. (2015) used interactive behavior trees (IBTs) to incorporate free-form user interaction and conflict resolution. Decision trees can be used to generate believable narratives, but they cannot practically support free-form user interaction.

To reduce the planning time, we introduced a hueristic–related actions, which will be acquired from the action relation map. The action relation map contains all the related actions for a condition. Thus it decreases the plan space thereby decreasing the planning time. Inspired from the D* and partial order planner, we came up with an approach to incorporate dynamic characteristics to the planner.

The planning time costs using the improved dynamic planner are compared against those of a conventional partial planner with IBTs. The time of execution and the optimality of the solution are taken as the measures to quantify the benefits of the proposed solution.

Our planner resembles the concept of mediation (Harris and Young 2009, Robertson and Young 2014). However, the dissimilarity and correspondence is yet to be discussed.

Further development is required before the final publication of our work, which mainly involves two aspects: algorithm optimization and comprehensive evaluation. Besides, evaluations of our planners against other classic planners such that proposed by Younes and Simmons (2003) are necessary.

Related Work

There have been several works on manual authoring techniques for the narrative generation. Work of Loyall (1997) described predefined behaviors, thus even small changes to the narrative result in monolithic work. The structure of story graphs (Gordon et al. 2004) allows to author huge believable narratives with very little user interactions. In story graphs, the user interactions are facilitated in the form of choices at key points in the narrative.

Partial order planning is often used to automatically generate narratives. Poole and Mackworth (2011) described the basic algorithm of a partial order planner to generate partial plans. A comparative study between total order and partial order planners has been done by Minton, Bresina, and Drummond (1994). Likhachev et al. (2005) described techniques to incorporate Anytime Dynamic nature in total ordering planners.

On the other hand, Shoulson et al. (2011) showed how behavior trees could be parameterized to improve the modularity and re-usability of the narratives. This helps to reuse similar story arcs to generate rather complex narratives. Li, Lee-Urban, and Riedl (2012) involved learning script-like narrative knowledge from crowdsourcing to generate narratives. Magerko et al. (2004) explored design issues of constructing a plot, creating AI characters, and using a director in an interactive storytelling environment. For implementing free-form user interactions, Kapadia et al. (2015) modeled an architecture with behavior trees. It also provided a conflict resolution algorithm in IBTs. Kallmann and Thalmann (1999) described a framework where virtual objects could aid the user to accomplish a pre-programmed possible interaction. An event-centric framework for directing interactive narratives was shown by Shoulson et al. (2013).

Loyall (1997) designed Hap, an architecture for creating believable agents. Shoulson and Badler (2011) described a framework for mitigating individual agent complexity while retaining agent diversity. The framework of Kapadia et al. (2011) generated complicated behaviors between interacting actors in a user-authored scenario. Kapadia, Marshak, and Badler (2014) presented a modular and flexible platform for authoring purposeful human characters in a virtual environment. Based on the framework, an attempt (Inampudi et al. 2017) has been made to generate compelling narratives by synthesizing the memories of the agents.

Harris and Young (2009) proposed another plan-based narrative generation algorithm which calculates responses to user input in an anticipatory manner. Ware and Young (2011) proposed the Conflict Partial Order Causal Link (CPOCL) that marks certain steps in a plan as non-executed in order to preserve the conflicting subplans of all characters without damaging the causal soundness of the overall story.

A survey of various approaches to quantifying the interestingness of a narrative was provided by Li. (2015). The work of Pérez and Ortiz (2013) was one of the earliest attempts to measure the interestingness of a computer generated plot. It compares ideal story tension graphs to the one that generated plot to find the interestingness. Although the framework of Kartal, Koenig, and Guy (2014) can be used to quickly generate complex and believable narratives, the memory requirements increase exponentially as the number of possible actions increases.

The Planners

In the following sections, we have made an attempt to describe the implementation of our planner and the game UI. Below are some terms that we will be used to describe our system.

Affordance: An affordance a is an action that involves two objects: the executor and the bearer. In this paper, we will use terms "affordance" and "action" interchangeably. Every affordance has a set of pre-conditions and effects ($< \{\Phi\}, \{\Omega\} >$). Pre-conditions are required for the action to execute, and effects are the state changes produced by the action after execution. Every affordance in our system also has a parameterized behavior tree (Shoulson et al. 2011) associated with it.

Causal link: Two actions, a_1 and a_2, have a causal link l over a condition ϕ, if a precondition ϕ of a_2 is satisfied by executing a_1, i.e., if $\phi \in a_1 \leftarrow \{\Omega\}$ and $\phi \in a_2 \leftarrow \{\Phi\}$. An entry of a causal link is denoted as a triple $< a_1, \phi, a_2 >$. A causal link $< a_1, \phi, a_2 >$ is said to be running if action a_1 is running and a_2 is not yet executed.

Over-consistent link: A causal link l will be in the over-consistent state if the condition of the causal link is already present in the current state of the system.

Under-consistent link: A running causal link l is said to be under-consistent if the condition is negated by the current state. A user action negating a running causal link will result in an under-consistent link.

Original Partial Order Planner

The partial order planner proposed by Poole and Mackworth (2011) was implemented to evaluate the performance of ours.

The planner employs a non-deterministic algorithm that stores all the goal states in an agenda at the beginning and depletes it by one entry in each iteration and stores corresponding ordered actions and causal links with the constraints of the preconditions on that entry. The new action must happen before the removed agenda entry in the partial order. The algorithm terminates when the agenda is empty, and outputs the ordered action list, which is the partial plan.

Accelerated Partial Order Planner (APOP)

One of the major factors that influence the planning time of a partial planner is the plan space. To optimize the planning time, we created an action relation map M which will be used to search for the actions that satisfy a precondition. Two actions are related if the precondition of an action is present in the effects of the other. This is created only when a new affordance is added and is stored in the system memory. Hence, this reduces the search space for the planner and hence improves the planning time. A single entry of M looks like $< \phi, \{a\} >$. Algorithm 1 describes the approach we used to create the action relation map. The APOP has been proven to be efficient in some recent work (Inampudi et al. 2017).

Dynamic Partial Order Planner (DPOP)

The existing techniques either preemptively plan for user interactions or replan the narrative from scratch, and none of these approaches are efficient to update the narratives in real time to achieve a lifelike simulation as the number of objects and interactions is prolific. To tackle this issue, we propose an algorithm which incorporates the concepts of the D* algorithm (Likhachev et al. 2005) with POP.

Algorithm 1: Constructing the action relation map and getting an action from the action relation map

```
1  ConstructActionRealtions ()
2      foreach Φ ∈ A ← {Φ} do
3          foreach a ∈ A do
4              if Φ ∈ a ← {Ω} then
5                  M = M ∪ < Φ, a >
```

The heuristic we use for the search algorithm is the action relation map M introduced in the previous section.

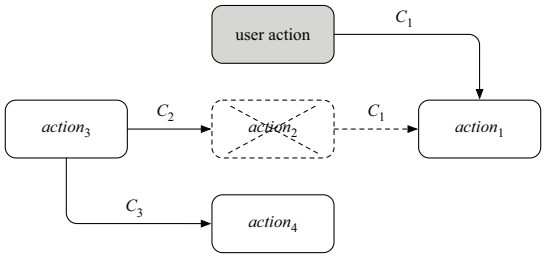


Figure 1: Example of an over-consistent link leading to redundant states

One of the improvements over the previous planner is removing redundant states. Over-consistent links (OC) may make some states in the narrative redundant, i.e., as is shown in figure 1, the user action (a_{ui}) has made the causal link C_1 over-consistent. Hence, $action_1$ is already satisfied by the user action, and $action_2$ becomes redundant, which is no longer required in the narrative. Removing redundant states from the narrative is called consistency propagation. Algorithm 2 describes the logic we employed to find and remove the redundant states from the narrative.

Algorithm 2: Consistency propagation in the behavior tree

```
1  PropagateConsistency(OC)
2      foreach a ∈ OC do
3          if a ∉ {L ← {a_1}} then
4              Remove a from affordances and constarints
5              foreach l ∈ L do
6                  if a ∈ l ← {a_2} then
7                      OC = OC ∪ l ← {a_1}
8      if OC ≠ ∅ then
9          PropagateConsistency(OC)
10     else
11         return
```

The planner keeps executing on open conditions to be satisfied. For every user action, the narrative state manager checks for any over-consistent and under-consistent states. If an over-consistent or under-consistent state is found, it is added to the open conditions of the planner. In the case of over-consistency, consistency propagation is performed before the repairing is done. See Algorithm 3 for more details.

One important feature of the planner DPOP is that it can repair or replan the narrative in real time when the user intervenes in the narrative. The algorithm to update the planner space accordingly (line 29 of algorithm 3) is shown in algorithm 4.

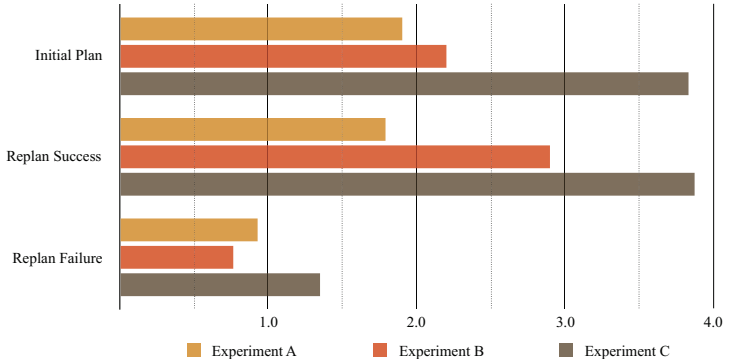
Proofs of the soundness and completeness of the planner can be found in the appendix.

Algorithm 3: The dynamic planner

```
1  begin
2      Φ_open = {< goal, φ > |∀φ ∈ goal ← {Φ}}
3      A = {start, goal}
4      O = {start ≺ goal}
5      L = ∅
6      initialPlanGenerated = false
7      repeat
8          if Φ_open ≠ ∅ then
9              < a_c, φ >= pop(Φ_open)
10             if a ← {Φ} ≠ φ ∀a ∈ A then
11                 a_s = SelectActionFromRelations(φ)
12                 A = A ∪ a_s
13                 O = O ∪ {start ≺ a_s}
14                 foreach l ∈ L do
15                     if l is not executed then
16                         O = Protect(l, a_s, O)
17                 foreach Φ ∈ a_s ← {Φ} do
18                     Φ_open = Φ_open ∪ {< a_s, Φ >}
19             else
20                 a_s = ∃a ∈ A s.t. φ ∈ a ← {Φ}
21             O = O ∪ {a_s ≺ a_c}
22             L = L ∪ {< a_s, φ, a_c >}
23             foreach a ∈ A do
24                 if a is not executed then
25                     O = Protect(< a_s, φ, a_c >, a, O)
26         else
27             initialPlanGenerated = true
28         if a_ui & initialPlanGenerated then
29             UpdatePannerSpace(a_ui)
30     until forever
```

Algorithm 4: Updating the planner space after user interaction a_{ui}

```
1  UpdatePlannerSpace(a_ui)
2      A = A ∪ a_ui
3      foreach l ∈ L do
4          if l ← φ ∈ a_ui ← {Ω} then
5              Remove l from L
6              OC = OC ∪ l ← {a_1}
7              foreach φ ∈ l ← a_2 ← {Φ} do
8                  Φ_open = Φ_open ∪ {< l ← a_2, φ >}
9      if OC ≠ ∅ then
10         PropagateConsistency(OC)
11     foreach rl ∈ RL do
12         if ¬rl ← φ ∈ a_ui ← {Ω} then
13             Remove rl from RL
14             foreach φ ∈ rl ← a_2 ← {Φ} do
15                 Φ_open = Φ_open ∪ {< rl ← a_2, φ >}
```



(a) Average performance boost of all experiments, represented as avg($\frac{\text{POP time}}{\text{DPOP time}}$)

Experiment	Observations	Standard deviation	Average	Confidence width	Min average	Max average
A: Initial plan	26	0.134	1.901	0.043	1.857	1.944
A: Replan success	31	1.077	1.776	0.318	1.457	2.094
A: Replan failure	22	0.602	0.921	0.021	0.900	0.942
B: Initial plan	23	0.539	2.253	0.185	2.068	2.438
B: Replan success	23	0.753	2.902	0.258	2.644	3.160
B: Replan failure	23	0.173	0.794	0.059	0.735	0.853
C: Initial plan	23	0.338	3.801	0.116	3.685	3.916
C: Replan success	23	0.880	3.841	0.302	3.539	4.143
C: Replan failure	23	0.779	1.394	0.267	1.127	1.661

(b) Performance boost statistics with 90% confidence, represented as avg($\frac{\text{POP time}}{\text{DPOP time}}$)

Experiment	Observations	Standard deviation	Average	Confidence width	Min average	Max average
A: Initial plan	26	0.134	1.900	0.051	1.849	1.952
A: Replan success	31	1.077	1.776	0.379	1.397	2.155
A: Replan failure	22	0.060	0.921	0.025	0.896	0.946
B: Initial plan	23	0.539	2.253	0.220	2.032	2.474
B: Replan success	23	0.752	2.902	0.308	2.594	3.209
B: Replan failure	23	0.172	0.794	0.071	0.724	0.865
C: Initial plan	23	0.338	3.801	0.138	3.663	3.939
C: Replan success	23	0.880	3.841	0.359	3.482	4.201
C: Replan failure	23	0.779	1.394	0.318	1.076	1.712

(c) Performance boost statistics with 95% confidence, represented as avg($\frac{\text{POP time}}{\text{DPOP time}}$)

Figure 2: Statistics of the experiments on DPOP with reference to the POP

Testing and Evaluation

The experiments were conducted within our game UI, which is the main experiment interface with the players.

About the Game UI

In our game UI, every smart object is associated with a smart trigger. When the player enters the trigger of the smart object, an interactive menu is shown with all the available affordances of that object. Selecting an affordance will execute the parameterized behavior tree associated with it, and also send the effects to the narrative state manager to update the corresponding states.

All the characters in the scene have a default behavior tree attached to them. The planner can take control of any character running on default behavior. The game allows the player to possess and control any in-game character, or switch among them while the experiment is in progress. Whenever the player takes control of a non-player character (NPC), all the behaviors on that NPC are suspended. If that NPC was formerly being controlled by a planner, the plan is recomputed. For every player action, the consistency of the plan is checked and repaired if necessary. Instead of taking over any character, the player can just observe the world from the character's point of view.

Experiment	Objects	Affordances/object	Plan space
A	4	≤ 3	12
B	23	≈ 8	2091
C	37	≈ 8	7257

Figure 3: Configurations for each experiment scene

Experiment Configuration

Our game is set up with scenes where the president assassinations occur. The planner is tested with three different scenes with a varying number of smart objects and affordances. The quantitative details of the experiments are shown in figure 3.

For each experiment, the data is collected for 3 cases: (i) initial plan; (ii) repairing or replanning success, when the planner is able to repair the narrative; (iii) repairing or replanning failure, when the planner is not able to repair the narrative.

Accelerated Planner

For initial tests, the APOP is run in a scene with a varying number of homogeneous objects, and the time costs are compared with the POP. The results are present in figure 4.

Objects	Affordances	Original POP	APOP
2	2	5.75	5.4
2	5	11.2	5.72
10	5	11.89	6.04

Figure 4: For homogeneous objects, average of 10 observations (time in milliseconds)

The initial planning case of the experiments corresponds to the APOP. The APOP displays significant improvements over the POP. The planning time efficiency improvement is exponential with regards to the total affordances in the scene, e.g., the third configuration that simulates a typical complex scene shows that the APOP is almost twice as fast as the POP.

Dynamic Planner

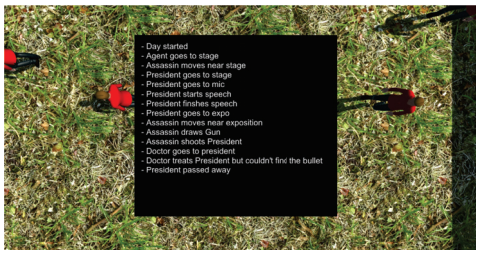
A detailed analysis is conducted for the experiments A, B and C. The data is split into two cases: case 1 when the DPOP tries to repair the plan succeeds, and case 2 when no plan exists between current state and goal state.

For case 1, the DPOP showed similar improvements as the APOP. The DPOP was as 3.7 to 3.9 times fast as POP with 90% confidence. For case 2, the DPOP is almost as fast as POP, which is expected because DPOP does an exhaustive search just like POP. Detailed analysis can be found in figure 2.

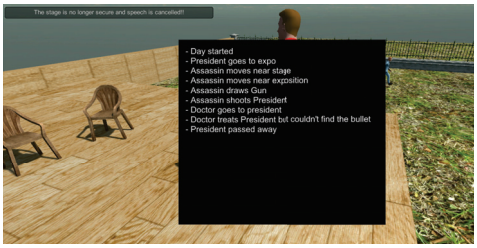
The following two figures show two demonstrating scenes where the DPOP works in.

Figure 5a and 5b reproduces a president assassination scene, where many types of agents are involved, including the player, the president, the assassin, and the doctor, etc. The initial narrative is generated and stored as the reference journal. While with the game UI interactions, the narrative is invalidated. With the DPOP fixing it, the complete case is revealed.

Figure 6a and 6b shows the scenes in experiment A where the DPOP succeeds and fails to repair the narrative in real time and interactively with the user. The initial goal of the narrative is that the NPC buys the weapon, but with user intervention that the player buys it before the NPC does, the narrative failed to repair, which is the intended result for the player–to prevent the NPC from purchasing the weapon.

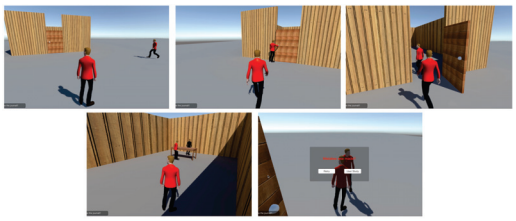


(a) The auto-generated initial narrative

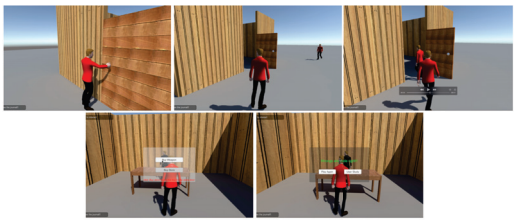


(b) The DPOP repairs the invalidated narrative

Figure 5: Screenshots from experiment C demonstrating a case where the narrative is repaired by DPOP



(a) Without user intervention, the NPC opens the door, walks in and buys the weapon



(b) The player opens the door, the NPC walks in without opening the door, then the player buys the weapon before the NPC does, and the narrative fails to be repaired

Figure 6: Two different cases of experiment A

User Study

In this user study, we did a quantitative analysis of the dynamic planner from the experimental results discussed in the previous section. An in-game questionnaire was conducted to comprise the analysis.

Hypotheses

There are two major hypotheses for our application:

1. The dynamic planner can reconstruct or repair the plan in real time and faster than a partial planner.

   Metrics: The time for replanning using POP and DPOP and the time for replanning from scratch using POP and DPOP.

2. This framework could be used to generate real-world like narratives with free-from interactions.

   Metrics: How close the simulation is to the original event and the complexity of the narrative.

DPOP analysis

The following metrics were collected from the game plays of 23 individuals. The game environment consists of 11 affordances and 4 heterogeneous smart objects.

The performance of the DPOP was analyzed with two planning types: (i) planning from scratch, and (ii) replanning (or repairing the plan). The corresponding individual experiment results are shown in figure 7 and figure 8.

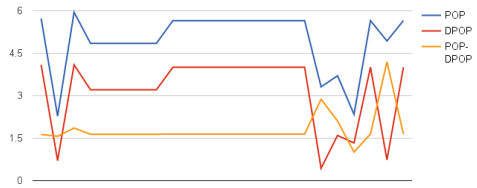


Figure 7: Time taken to generate a partial plan (in ms)

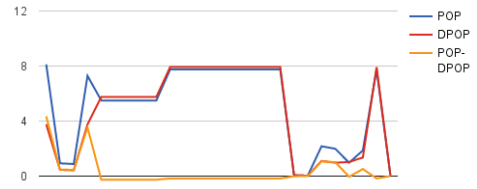


Figure 8: Time taken to repair the plan (in ms)

As per our user study results and the feedback from the testing users, we can come to the conclusion that:

1. The APOP and the DPOP have shown improvements over the POP according to the test data;
2. The DPOP is able to repair plans in real time, without much loss of user experience.

Conclusion and Future Work

Compared with the POP, our APOP and DPOP shows adequate performance improvements. With the DPOP, we can generate and repair the narrative in real time with free-form user interactions. It also shows the capability to recreate or simulate the possible variants of real world events, and potentially provides an extension for creative scenarios, e.g., replaying historic events.

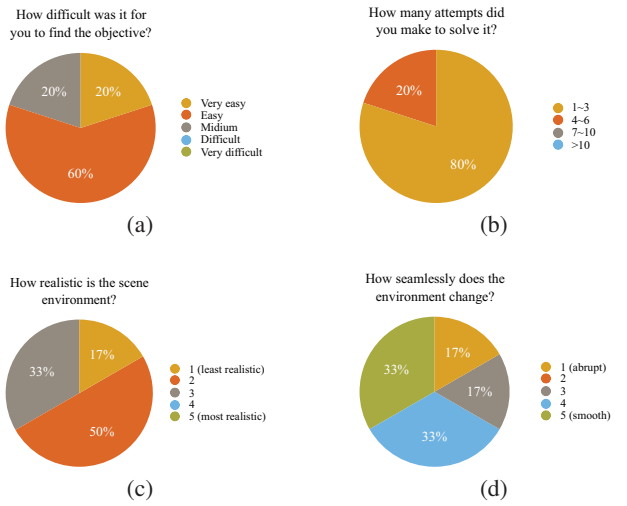


Figure 9: Player responses for the user study questionnaire

Though the system used in this paper facilitates to prove the working and efficiency of the APOP and the DPOP, it could only do so for a predetermined number of affordances and smart objects. The user study also implies that the realism of the testing environment is not outstanding. To accomplish a more comprehensive evaluation than those discussed previously, we need to create a complex scene and going to conduct a more perplex narrative experiment.

To extend the universality of our system, we need to implement a system that can procedurally generate smart objects and affordances so that universal statistics of a planner can be measured. The output (narrative) quality of the planner can be improved by using better heuristics like interestingness or tension induced in the story if an action is added.

Additionally, although we have tested our planners within various scenes, the performance was not measured against other types of planners due to the limited time for interface programming. Deeper studies of comparison among these planners shall be conducted.

Appendix: Proofs

In this section we have made an attempt to prove the soundness and completeness of the planner.

Proposition 1 (Soundness)**.** *For a solution plan* $\pi_c = \textbf{\textit{DynamicPlan}}(\pi_p)$*, there exists a total ordering of event instances to produce a complete and consistent story arc* α_c*.*

Proof. The successful termination condition of **DynamicPlan** returns a plan π_c that has no open preconditions, whose ordering constraints are guaranteed to be consistent, and there are no incomplete event instances with missing parameters. According to Kapadia et al. (2016), π_c is a solution plan and hence, there exists a total ordering of event instances α_c for π_c. □

Proposition 2 (Completeness)**.** ***DynamicPlan*** *is guaranteed to return a complete, consistent ordering of event instances π_c, if one exists.*

Proof. **DynamicPlan** is a non-deterministic procedure and we can prove completeness by showing that at least one of its execution traces returns a solution, if one exists. Let us prove it inductively on the number of event instances $|\mathrm{I}|$ in the solution plan.
Base Step. If the original plan π_p is a solution to the problem, then **DynamicPlan** terminates immediately and returns π_p.
Induction Step. Assume that **DynamicPlan** is complete for planning problems that have solutions of length $(k-1)$.

For a planning problem $\langle \pi_p = \langle \mathrm{I}_p, \mathrm{O}_p, \mathrm{L}_p \rangle, \mathrm{A} \rangle$, let there be a solution π_c which can be obtained by adding k additional event instances $\mathrm{I}_k = \{I_1, \cdots I_k\}$ to I_p. Since $I_k \in \mathrm{I}_k$ is relevant to satisfying a clause in A, there is at least one execution trace of **DynamicPlan** that chooses I_k to address an open clause in A. The resulting partial plan, $\pi^{'}$ contains $\mathrm{I}^{'} = \mathrm{I}_p \cup I_k$. The next recursion step of **DynamicPlan** operates on $\pi^{'}$ to generate a solution for the planning problem: $\langle \pi^{'}, \mathrm{A}' = \mathrm{A} \cup \Phi(I_k) \rangle$, whose solution can be obtained by adding $(k-1)$ events, $\mathrm{I}_{k-1} = \mathrm{I}_k - I_k$, which is our starting assumption. By induction, **DynamicPlan** has an execution trace that finds a solution for $\pi^{'}$. □

References

Gordon, A. S.; van Lent, M.; van Velson, M.; Carpenter, P.; and Jhala, A. 2004. Branching Storylines in Virtual Reality Environments for Leadership Development. In *Proceedings of the 16th Innovative Applications of Artificial Intelligence Conference (IAAI-04)*, 844–851. San Jose, CA: AAAI Press.

Harris, J., and Young, R. M. 2009. Proactive mediation in plan-based narrative environments. In *IEEE Transactions on Computational Intelligence and AI in Games*, volume 1, 233–244.

Inampudi, B. C.; Zhang, X.; Geraci, F.; Badler, N. I.; and Kapadia, M. 2017. Memory reconstruction from autobiographic memories of autonomous virtual agents. In *Proceedings of the 30th International Conference on Computer Animation and Social Agents (CASA)*, 79–88.

Kallmann, M., and Thalmann, D. 1999. Direct 3d interaction with smart objects. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '99, 124–130. New York, NY, USA: ACM.

Kapadia, M.; Singh, S.; Reinman, G.; and Faloutsos, P. 2011. A behavior-authoring framework for multiactor simulations. *Computer Graphics and Applications, IEEE* 31(6):45 –55.

Kapadia, M.; Falk, J.; Zund, F.; Marti, M.; Sumner, R. W.; and Gross, M. 2015. Computer-assisted authoring of interactive narratives.

Kapadia, M.; Frey, S.; Shoulson, A.; Sumner, R. W.; and Gross, M. 2016. CANVAS: Computer-Assisted Narrative Animation Synthesis. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '16. Eurographics.

Kapadia, M.; Marshak, N.; and Badler, N. I. 2014. ADAPT: The Agent Development and Prototyping Testbed. *IEEE Transactions on Visualization and Computer Graphics* 99(PrePrints):1.

Kartal, B.; Koenig, J.; and Guy, S. J. 2014. User-driven narrative variation in large story domains using monte carlo tree search. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, 69–76. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Li, B.; Lee-Urban, S.; and Riedl, M. O. 2012. Crowdsourcing narrative intelligence. *Advances in Cognitive Systems* 2.

Li., B. 2015. *Learning Knowledge to Support Domain-Independent Narrative Intelligence*. Ph.D. Dissertation.

Likhachev, M.; Ferguson , D.; Gordon, G.; Stentz , A. T.; and Thrun, S. 2005. Anytime dynamic a*: An anytime, replanning algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Loyall, A. B. 1997. *Believable agents: building interactive personalities*. Ph.D. Dissertation, Pittsburgh, PA, USA.

Magerko, B.; Laird, J. E.; Assanie, M.; Kerfoot, A.; and Stokes, D. 2004. AI Characters and Directors for Interactive Computer Games. *Artificial Intelligence* 1001:877–883.

Minton, S.; Bresina, J. L.; and Drummond, M. 1994. Total-order and partial-order planning: A comparative analysis. *CoRR* abs/cs/9412103.

Pérez, R. P., and Ortiz, O. 2013. A model for evaluating interestingness in a computer–generated plot,. In *Proceedings of the Fourth International Conference on Computational Creativity*, 131–138.

Poole, D. L., and Mackworth, A. K. 2011. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press.

Robertson, J., and Young, R. M. 2014. Gameplay as on-line mediation search. In *Experimental Artificial Intelligence in Games*, 42–28.

Shoulson, A., and Badler, N. I. 2011. Event-centric control for background agents. In *ICIDS*, 193–198.

Shoulson, A.; Garcia, F. M.; Jones, M.; Mead, R.; and Badler, N. I. 2011. Parameterizing behavior trees. In *4th International Conference, MIG 2011, Edinburgh, UK, November 13-15, 2011. Proceedings*.

Shoulson, A.; Gilbert, M. L.; Kapadia, M.; and Badler, N. I. 2013. An event-centric planning approach for dynamic real-time narrative. In *Proceedings of Motion on Games*, MIG '13, 99:121–99:130. New York, NY, USA: ACM.

Ware, S. G., and Young, R. M. 2011. Cpocl: A narrative planner supporting conflict. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 97–102.

Younes, H. L., and Simmons, R. G. 2003. Vhpop: Versatile heuristic partial order planner. In *Journal of Artificial Intelligence Research 20*, 405–430.